

1 **In the Claims**

2 Claims 1, 13, 14, 37, 71 and 94 are amended.

3 Claims 62 and 63 were previously canceled.

4 Claims 1-61 and 64-96 remain in the application and are listed as follows:

5
6 1. (Currently Amended) A method for delivering software via a
7 network comprising:

8 describing one or more software extensions using a hierarchical language,
9 the extensions being configured for incorporation on a client, said describing
10 defining one or more manifests containing at least one list of files comprising an
11 extension; and

12 delivering the one or more manifests to the client via the network, the one
13 or more manifests being configured for use in downloading the software
14 extensions via the network, at least some of the extensions being downloadable by
15 streaming extension files to the client in a manner that enables a user to begin to
16 interact with the extension sooner than if the user had to wait for the entire
17 extension to load, said manner being developed based on scenario runs in which
18 files that are more likely to be first used by the user are downloaded before files
19 that are less likely to be first used, and wherein files that are less likely to be used
20 first can be downloaded via a background download process.

21
22 2. (Original) The method of claim 1, wherein the one or more
23 manifests are configured to assist in organizing delivery of individual files listed in
24 the one or more manifests.
25

1 3. (Original) The method of claim 1, wherein the one or more
2 manifests are configured to assist in validating individual files listed in the one or
3 more manifests.

4
5 4. (Original) The method of claim 1, wherein the one or more
6 manifests are configured to assist in updating individual files listed in the one or
7 more manifests.

8
9 5. (Original) The method of claim 1, wherein the one or more
10 manifests describe individual file locations.

11
12 6. (Original) The method of claim 1, wherein the one or more
13 manifests contain individual hashes for one or more of the listed files.

14
15 7. (Original) The method of claim 1, wherein the one or more
16 manifests contain download directives for downloading the listed files.

17
18 8. (Original) The method of claim 1, wherein the one or more
19 manifests are defined in a tag-based language.

20
21 9. (Original) The method of claim 1, wherein the one or more
22 manifests are defined in extensible markup language (XML).

23
24 10. (Original) The method of claim 1, wherein the network comprises
25 the Internet.

1
2 11. (Original) One or more computer-readable media comprising
3 computer-readable instructions thereon which, when executed by a computer,
4 cause the computer to implement the method of claim 1.
5

6 12. (Original) A computer programmed with instructions which, when
7 executed by the computer, implemented the method of claim 1.
8

9 13. (Currently Amended) One or more computer-readable media
10 comprising computer-readable instructions thereon which, when executed by a
11 computer, cause the computer to:

12 describe one or more software extensions using extensible markup
13 language (XML), the extensions being configured for incorporation on a client,
14 said describing defining a manifest containing at least one list of files comprising
15 an extension, the manifest being configured to assist in one or more of the
16 following: organizing delivery of individual files listed in the manifest, validating
17 individual files listed in the manifest, and updating individual files listed in the
18 manifest; and

19 deliver the manifest to the client via the network, at least some of the
20 extensions being downloadable by streaming extension files to the client in a
21 manner that enables a user to begin to interact with the extension sooner than if the
22 user had to wait for the entire extension to load, said manner being developed
23 based on scenario runs in which files that are more likely to be first used by the
24 user are downloaded before files that are less likely to be first used, and wherein
25

1 files that are less likely to be used first can be downloaded via a background
2 download process.

3
4 14. (Currently Amended) A method for receiving software via a network
5 comprising:

6 receiving a manifest that contains at least one list of files comprising a
7 software extension that is to be downloaded via a network and incorporated on a
8 client, the manifest being defined in extensible markup language (XML), the
9 manifest being configured to assist in:

10 organizing delivery of the files,

11 validating individual files listed in the manifest, and

12 updating individual files listed in the manifest; and

13 downloading files from the list of files contained in the manifest;

14 wherein the extension is downloadable by streaming extension files to the
15 client in a manner that enables a user to begin to interact with the extension sooner
16 than if the user had to wait for the entire extension to load, said manner being
17 developed based on scenario runs in which files that are more likely to be first
18 used by the user are downloaded before files that are less likely to be first used,
19 and wherein files that are less likely to be used first can be downloaded via a
20 background download process.

21
22 15. (Original) The method of claim 14, wherein the software extension
23 is to be incorporated into a software platform executing on the client.
24
25

1 16. (Original) The method of claim 14, wherein the downloading of the
2 files takes place by downloading the files in an order that is described in the
3 manifest.

4
5 17. (Original) The method of claim 14 further comprising validating one
6 or more downloaded files using the manifest.

7
8 18. (Original) The method of claim 14 further comprising updating one
9 or more files using the manifest.

10
11 19. (Original) One or more computer-readable media comprising
12 computer-readable instructions thereon which, when executed by a computer,
13 cause the computer to implement the method of claim 14.

14
15 20. (Original) A computer programmed with instructions which, when
16 executed by the computer, implement the method of claim 14.

17
18 21. (Previously Presented) A data structure embodied on a computer-
19 readable medium, the data structure comprising:

20 a list of one or more files that are utilized in a software extension that is
21 configured to extend a software application executing on a client;

22 one or more hashes each of which being associated with a particular listed
23 file; and

1 one or more file groups, individual files being associated with individual
2 file groups, the file groups determining when particular files of the extension get
3 downloaded to the client;

4 the data structure being configured to assist in delivering software
5 extensions via the Internet.
6

7 22. (Original) The data structure of claim 21, wherein the file groups
8 determine where files are stored on the client.
9

10 23. (Original) The data structure of claim 21, wherein the file groups
11 determine how files are packaged for delivery.
12

13 24. (Original) The data structure of claim 21, wherein the file groups
14 determine where files are stored on the client and how files are packaged for
15 delivery.
16

17 25. (Original) The data structure of claim 21, wherein the file groups
18 identify files that are to be downloaded before any other files.
19

20 26. (Original) The data structure of claim 21, wherein the file groups
21 identify files that are to be downloaded for offline use.
22

23 27. (Original) The data structure of claim 21, wherein the file groups
24 identify files that are only downloaded when they are required for the first time by
25 a user.

1
2 28. (Original) The data structure of claim 21, wherein the file groups
3 identify files that are downloaded on demand and provide content that is available
4 only when a user is online.
5

6 29. (Original) The data structure of claim 21, wherein the file groups
7 indicate file download priority.
8

9 30. (Original) The data structure of claim 21, wherein the one or more
10 hashes are used for security.
11

12 31. (Original) The data structure of claim 21, wherein the one or more
13 hashes are used for versioning.
14

15 32. (Original) The data structure of claim 21, further comprising a
16 storage size that is associated with an amount of storage required by the extension.
17

18 33. (Original) The data structure of claim 21, further comprising one or
19 more class identifiers for individual dynamic link libraries (DLLs).
20

21 34. (Original) The data structure of claim 21, further comprising one or
22 more dynamic link library (DLL) load dependencies.
23

24 35. (Original) The data structure of claim 21, further comprising:
25

1 a storage size that is associated with an amount of storage required by the
2 extension;

3 one or more class identifiers for individual dynamic link libraries (DLLs);
4 and

5 one or more dynamic link library (DLL) load dependencies.

6
7 36. (Original) The data structure of claim 21 embodied as an extensible
8 markup language (XML) file.

9
10 37. (Currently Amended) A method of providing software via a network
11 comprising:

12 describing one or more software extensions using one or more extensible
13 markup language (XML) files, the extensions being configured for incorporation
14 in a software program executing on a client, individual XML files providing
15 individual manifests that contain a list of files that comprise an extension; and

16 storing the XML files in a Web-accessible location;

17 wherein at least some of the extensions are downloadable by streaming
18 extension files to the client in a manner that enables a user to begin to interact with
19 the extension sooner than if the user had to wait for the entire extension to load,
20 said manner being developed based on scenario runs in which files that are more
21 likely to be first used by the user are downloaded before files that are less likely to
22 be first used, and wherein files that are less likely to be used first can be
23 downloaded via a background download process.

1 38. (Original) The method of claim 37 further comprising storing
2 extension files associated with the XML files in a Web-accessible location.

3
4 39. (Original) The method of claim 37 further comprising:
5 storing extension files associated with the XML files in a Web-accessible
6 location; and

7 providing one or more XML files and one or more associated extension
8 files to a client via the network.

9
10 40. (Original) The method of claim 37, wherein individual manifests can
11 contain one or more file hashes that can be used for file security.

12
13 41. (Original) The method of claim 37, wherein individual manifests can
14 contain one or more file hashes that can be used for versioning.

15
16 42. (Original) The method of claim 37, wherein individual manifests
17 comprise one or more file groups that determine when particular files are
18 downloaded to the client.

19
20 43. (Original) The method of claim 42, wherein the file groups
21 determine where files are stored on the client.

22
23 44. (Original) The method of claim 42, wherein the file groups
24 determine how files are packaged.

1 45. (Original) The method of claim 42, wherein the file groups
2 determine where files are stored on the client and how files are packaged.

3
4 46. (Original) One or more computer-readable media having computer-
5 readable instructions thereon which, when executed by a computer, cause the
6 computer to implement the method of claim 37.

7
8 47. (Original) A security method for downloading software extensions
9 via the Internet comprising:

10 receiving, via the Internet, a package manifest containing a list of multiple
11 files that comprise a software extension that is to be incorporated into an
12 application program executing on a client, the list containing a hash for one or
13 more of the files comprising the software extension;

14 receiving, via the Internet, the multiple files that are described in the
15 package manifest;

16 creating a hash for one or more of the multiple received files; and

17 comparing the created hash of the one or more files with corresponding file
18 hashes contained in the package manifest to ascertain whether one or more of the
19 received file is secure.

20
21 48. (Original) The security method of claim 47, wherein the package
22 manifest comprises an XML file.

1 49. (Original) One or more computer-readable media comprising
2 computer-readable instructions thereon which, when executed by a computer,
3 cause the computer to implement the method of claim 47.

4
5 50. (Original) One or more client computers programmed with
6 instructions which, when executed by the one or more computers, cause the one or
7 more computers to implement the method of claim 47.

8
9 51. (Original) An updating method for updating software extensions via
10 the Internet comprising:

11 receiving, via the Internet, a package manifest containing a list of multiple
12 files that comprise a newer version of a software extension that is to be
13 incorporated into an application program executing on a client that contains an
14 older software extension version, the list containing a hash for one or more of the
15 files comprising the newer version of the software extension;

16 comparing one or more hashes that are received with one or more hashes of
17 files from the older version of the software extension;

18 for any hashes of corresponding files from the different versions that are
19 different, downloading a new file from a web server; and

20 for any hashes of corresponding files from the different versions that are the
21 same, copying a file from an old local directory on the client to a new local
22 directory on the client associated with the newer version of the extension.

23
24 52. (Original) The updating method of claim 51, wherein the package
25 manifest comprises an XML file.

1
2 53. (Original) One or more computer-readable media comprising
3 computer-readable instructions thereon which, when executed by a computer,
4 cause the computer to implement the method of claim 51.
5

6 54. (Original) One or more client computers programmed with
7 instructions which, when executed by the one or more computers, cause the one or
8 more computers to implement the method of claim 51.
9

10 55. (Original) A data structure embodied on a computer-readable
11 medium comprising:

12 one or more first tags indicative of associated file groups associated with an
13 Internet-downloadable software extension that can extend an application program
14 executing on a client; and

15 one or more second tags indicative of specific files that comprise the
16 software extension.
17

18 56. (Original) The data structure of claim 55, wherein the one or more
19 second tags can contain hashes for individual files.
20

21 57. (Original) The data structure of claim 55 further comprising one or
22 more third tags indicative of file load dependencies.
23

24 58. (Original) The data structure of claim 55 further comprising one or
25 more third tags indicative of COMClass IDs.

1
2 59. (Original) The data structure of claim 55 further comprising:
3 one or more third tags indicative of file load dependencies; and
4 one or more fourth tags indicative of COMClass IDs.
5

6 60. (Original) The data structure of claim 55, wherein the ordering of the
7 file groups implicitly defines a download order of the files.
8

9 61. (Original) The data structure of claim 55, wherein the tags comprise
10 XML tags.
11

12 62. (Cancelled).
13

14 63. (Cancelled).
15

16 64. (Original) A queue management method comprising:
17 defining a download queue that controls when files are to be downloaded to
18 a client, the files pertaining to a software extension that is to be incorporated into
19 an application program executing on the client;

20 ascertaining whether a user action at the client requires one or more files
21 that are not currently being downloaded; and

22 manipulating the download queue responsive to a user action that requires
23 one or more files that are not currently being downloaded so that the one or more
24 required files are downloaded sooner than they would otherwise be.
25

1 65. (Original) The queue management method of claim 64, wherein the
2 download queue contains multiple package objects each of which contains a list of
3 one or more files that correspond to a software extension, and said manipulating
4 comprises moving a package object to the head of the download queue.

5
6 66. (Original) The queue management method of claim 65, wherein said
7 ascertaining comprises determining whether a required file is associated with a
8 package object whose files are currently being downloaded.

9
10 67. (Original) The queue management method of claim 65, wherein said
11 manipulating comprises:

12 ascertaining an identifier associated with a requested file; and

13 locating a package object with a corresponding identifier.

14
15 68. (Original) One or more computer-readable media comprising
16 computer-readable instructions thereon which, when executed by a computer,
17 cause the computer to implement the method of claim 64.

18
19 69. (Previously Presented) A method of creating software packages for
20 delivery via the Internet comprising:

21 identifying end user features;

22 identifying shared dependencies between the end user features;

23 creating individual software packages for the end user features;

24 creating individual software packages for the shared dependencies; and

25 hosting the software packages on a web server;

1 wherein both of said acts of identifying and both of said acts of creating
2 provide software extensions that are created in a uniform manner independent of
3 end user input.
4

5 70. (Original) The method of claim 69, wherein the packages comprise
6 one or more files that are associated with a single application program that
7 provides multiple different functionalities, and the packages extend, in some way,
8 the functionalities that are provided by the single application program.
9

10 71. (Currently Amended) An automated software tool comprising a
11 package manifest creation tool configured to:

12 receive one or more input parameters pertaining to a package manifest that
13 is to describe a software extension that is configured to extend a software
14 application executing on a client; and

15 generate a package manifest that describes the extension, the package
16 manifest being generated using a hierarchical language;

17 wherein the extension is downloadable by streaming extension files to the
18 client in a manner that enables a user to begin to interact with the extension sooner
19 than if the user had to wait for the entire extension to load, said manner being
20 developed based on scenario runs in which files that are more likely to be first
21 used by the user are downloaded before files that are less likely to be first used,
22 and wherein files that are less likely to be used first can be downloaded via a
23 background download process.
24
25

1 72. (Original) The automated software tool of claim 71, wherein the
2 hierarchical language comprises a tag-based language.

3
4 73. (Original) The automated software tool of claim 71, wherein the
5 hierarchical language comprises extensible markup language (XML).

6
7 74. (Original) The automated software tool of claim 71, wherein one
8 input parameter specifies a directory containing files that are to be described in the
9 package manifest.

10
11 75. (Original) The automated software tool of claim 71, wherein one
12 input parameter comprises file group information and load dependencies.

13
14 76. (Original) The automated software tool of claim 71, wherein one
15 input parameter comprises file usage statistics.

16
17 77. (Original) The automated software tool of claim 76, wherein the file
18 usage statistics are ascertained from scenario runs.

19
20 78. (Original) The automated software tool of claim 77, wherein
21 individual scenarios have individual priorities.

22
23 79. (Original) The automated software tool of claim 76, wherein the file
24 usage statistics are ascertained from scenario runs that are collected by running
25 logs on various scenarios.

1
2 80. (Original) The automated software tool of claim 76, wherein the file
3 usage statistics are ascertained from scenario runs that are collected by running
4 logs on various scenarios, the scenarios having individual checkpoints that
5 separate one scenario from another.
6

7 81. (Previously Presented) The automated software tool of claim 76,
8 wherein the file usage statistics are ascertained dynamically by building a
9 knowledge base that describes tasks that users typically accomplish.
10

11 82. (Original) A method for creating a package manifest comprising:
12 receiving information pertaining to one or more extension directories that
13 contain files that comprise a software extension that is to be incorporated into a
14 software application program to extend the application program;
15 receiving, if any, information pertaining to file groups or load
16 dependencies;
17 receiving, if any, information pertaining to file usage statistics; and
18 generating a package manifest based on the received information.
19

20 83. (Original) The method of claim 82, wherein the package manifest is
21 generated in XML.
22

23 84. (Original) The method of claim 82, wherein the file usage statistics
24 can be provided based upon scenario runs.
25

1 85. (Original) One or more computer-readable media having computer-
2 readable instructions thereon which, when executed by a computer, cause the
3 computer to implement the method of claim 82.

4
5 86. (Original) A method of providing software extensions via the
6 Internet comprising:

7 assigning one or more files to one or more scenarios to provide multiple
8 different scenarios that describe ways that a user interacts with a software
9 application program;

10 assigning a priority to each of the scenarios;

11 sorting multiple files in accordance with their scenario priority or priorities;

12 and

13 downloading sorted files in an order defined by said sorting.

14
15 87. (Original) The method of claim 86 further comprising sorting the
16 multiple files in accordance with one or more file groups.

17
18 88. (Original) The method of claim 86 further comprising sorting the
19 multiple files in accordance with a file usage order within one or more scenarios.

20
21 89. (Original) The method of claim 86 further comprising:
22 sorting the multiple files in accordance with one or more file groups; and
23 sorting the multiple files in accordance with a file usage order within one or
24 more scenarios.

25

1 90. (Original) One or more computer-readable media having computer-
2 readable instructions thereon which, when executed by a computer, cause the
3 computer to implement the method of claim 86.

4
5 91. (Original) A method of ordering files for download to a client
6 comprising:

7 sorting multiple files by one or more file groups;

8 sorting the multiple files based on scenario priority of one or more
9 scenarios into which each file can be placed;

10 sorting the multiple files by file usage order within one or more scenarios.

11
12 92. (Original) The method of claim 91 further comprising determining a
13 file group from a manifest defined in XML.

14
15 93. (Original) The method of claim 91 further comprising if no file
16 group information is provided, assuming that all files of a particular type are of a
17 higher priority than other files of different types.

18
19 94. (Currently Amended) The method of claim 91, wherein said sorting
20 by file usage order comprises sorting the files according to the average order in
21 which the files were downloaded within scenarios of their particular priority or
22 priorities.

1 95. (Original) One or more computer-readable media having computer-
2 readable instructions thereon which, when executed by a computer, cause the
3 computer to implement the method of claim 91.

4
5 96. (Original) One or more server computers programmed with
6 instructions which, when executed by the one or more server computers, cause the
7 one or more server computers to implement the method of claim 91.